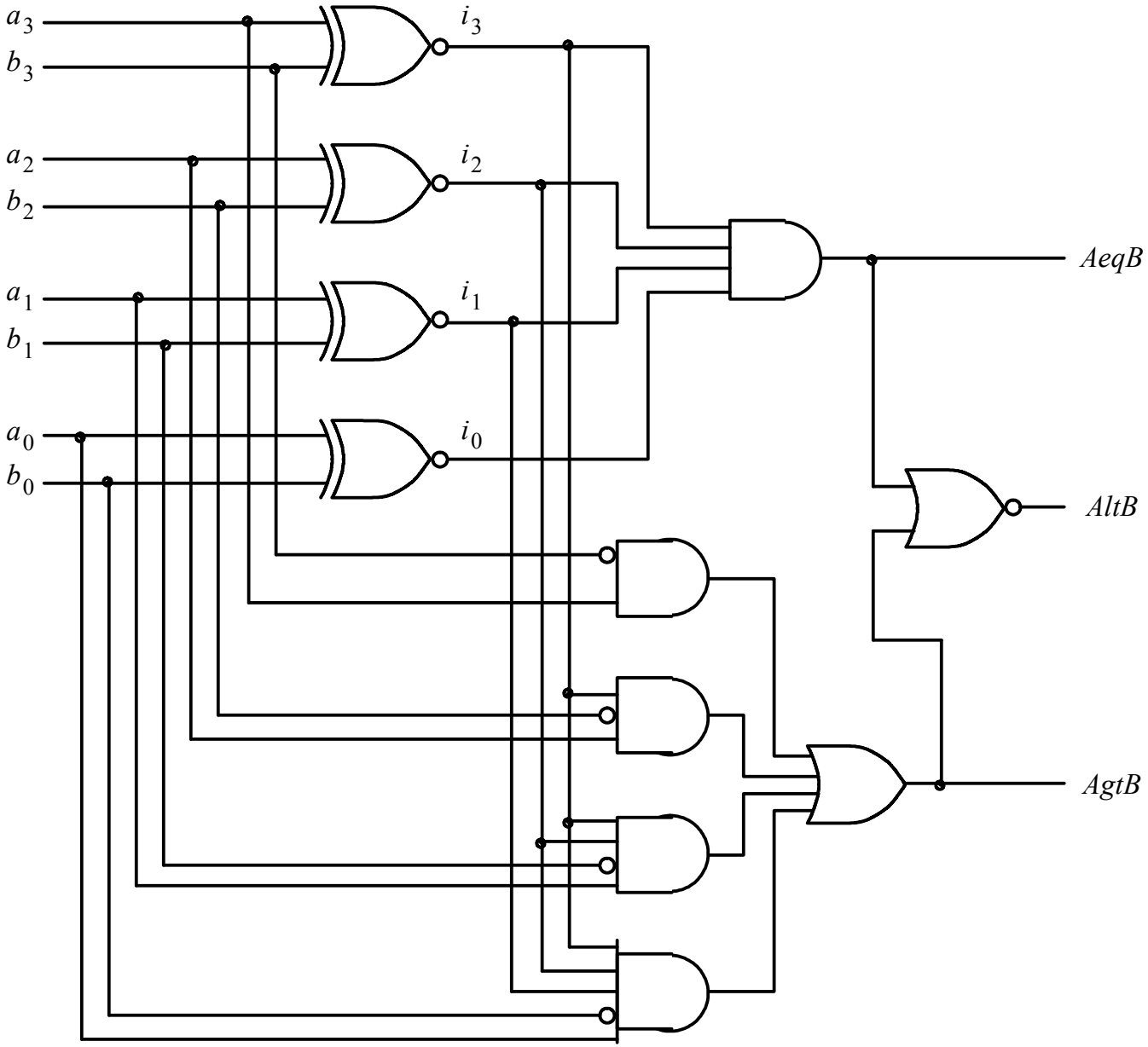
IC-UNICAMP

MC 613

IC/Unicamp
2013s1
Prof Guido Araújo
Prof Mario Côrtes

Circuitos Combinacionais Típicos (continuação)

Comparador de 4 bits



IC-UNICAMP

Comparador de 4 bits – VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY compare IS
   PORT (A, B: IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
         AeqB, AgtB, AltB  : OUT    STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
   AeqB <= '1' WHEN A = B ELSE '0' ;
   AgtB <= '1' WHEN A > B ELSE '0' ;
   AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

IC-UNICAMP

Comparador de 4 bits (signed) – VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.std_logic_arith.all ;

ENTITY compare IS
   PORT (A, B: IN    SIGNED(3 DOWNTO 0) ;
         AeqB, AgtB, AltB  : OUT    STD_LOGIC ) ;
END compare ;

ARCHITECTURE Behavior OF compare IS
BEGIN
   AeqB <= '1' WHEN A = B ELSE '0' ;
   AgtB <= '1' WHEN A > B ELSE '0' ;
   AltB <= '1' WHEN A < B ELSE '0' ;
END Behavior ;
```

IC-UNICAMP

Números com sinal

- **`PORT (A, B: IN SIGNED(3 DOWNTO 0)`**
- Necessita da biblioteca
  - **`USE ieee.std_logic_arith.all`**
- Qual é o efeito nos valores de **`AeqB, AgtB, AltB?`**

IC-UNICAMP

Codificador de prioridade – VHDL

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
   PORT (w  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
         y  : OUT       STD_LOGIC_VECTOR(1 DOWNTO 0) ;
         z  : OUT       STD_LOGIC ) ;
END priority ;

ARCHITECTURE Behavior OF priority IS
BEGIN
   y <= "11" WHEN w(3) = '1' ELSE
        "10" WHEN w(2) = '1' ELSE
        "01" WHEN w(1) = '1' ELSE
        "00" ;
   z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

IC-UNICAMP

Codificador de prioridade – ineficiente (1)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
   PORT ( w: IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
          y : OUT   STD_LOGIC_VECTOR(1 DOWNTO 0) ;
          z : OUT   STD_LOGIC ) ;
END priority ;
```

IC-UNICAMP

Codificador de prioridade – ineficiente (2)

```
ARCHITECTURE Behavior OF priority IS
BEGIN
   WITH w SELECT
      y <="00" WHEN "0001",
           "01" WHEN "0010",
           "01" WHEN "0011",
           "10" WHEN "0100",
           "10" WHEN "0101",
           "10" WHEN "0110",
           "10" WHEN "0111",
           "11" WHEN OTHERS ;
   WITH w SELECT
      z <=      '0' WHEN "0000",
           '1' WHEN OTHERS ;
END Behavior ;
```

IC-UNICAMP

MUX 16:1 com GENERATE (1)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE work.mux4to1_package.all ;

ENTITY mux16to1 IS
    PORT (   w      : IN  STD_LOGIC_VECTOR(0 TO 15) ;
             s   : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
             f   : OUT       STD_LOGIC ) ;
END mux16to1 ;
```

IC-UNICAMP

MUX 16:1 com GENERATE (2)

```
ARCHITECTURE Structure OF mux16to1 IS
    SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;

BEGIN
    G1: FOR i IN 0 TO 3 GENERATE
       Muxes: mux4to1 PORT MAP (
          w(4*i), w(4*i+1), w(4*i+2),
          w(4*i+3), s(1 DOWNTO 0), m(i) ) ;
    END GENERATE ;

    Mux5: mux4to1 PORT MAP
          ( m(0), m(1), m(2), m(3), s(3 DOWNTO 2), f ) ;
END Structure ;
```

Decodificador 4:16 – projeto hierárquico (1)

IC-UNICAMP

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY dec4to16 IS
   PORT (   w  : IN     STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            En : IN     STD_LOGIC ;
            y  : OUT  STD_LOGIC_VECTOR(0 TO 15) ) ;
END dec4to16 ;

ARCHITECTURE Structure OF dec4to16 IS
   COMPONENT dec2to4
      PORT (   w  : IN       STD_LOGIC_VECTOR(1 DOWNTO 0)
               En : IN     STD_LOGIC ;
               y  : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
   END COMPONENT ;
```

IC-UNICAMP

Decodificador 4:16 – projeto hierárquico (2)

```
    SIGNAL m : STD_LOGIC_VECTOR(0 TO 3) ;
BEGIN
    G1: FOR i IN 0 TO 3 GENERATE
        Dec_ri: dec2to4 PORT MAP
                ( w(1 DOWNTO 0), m(i), y(4*i TO 4*i+3) );
        G2: IF i=3 GENERATE
            Dec_left: dec2to4 PORT MAP
                ( w(i DOWNTO i-1), En, m ) ;
        END GENERATE ;
    END GENERATE ;
END Structure ;
```

IC-UNICAMP

VHDL: comandos sequenciais

- Visto até agora: comandos concorrentes
  - Ordem entre os comandos não importa
  - Analogia com componentes eletrônicos
- Novo conceito: process

```
C <= D and E;
PROCESS ( A, B )
  VARIABLE  x: STD_LOGIC
  BEGIN
       ……..    -- corpo do processo
  END PROCESS ;
E <= A or B;
```

IC-UNICAMP

Algumas características de processo

```
C <= D and E;
PROCESS ( A, B )
   VARIABLE  x: STD_LOGIC
   BEGIN
          ……..-- corpo do processo
   END PROCESS ;
E <= A or B;
```

- Trecho entre Begin e End é executado sequencialmente (a ordem importa)
- O processo é executado concorrentemente como as demais declarações (3 comandos concorrentes no exemplo)
- O processo é invocado quando muda algum sinal/variável na lista de sensibilidade (A,B)
- VARIABLE: possível somente dentro de processos
  - Atribuição x := ‘1’
  - Escopo somente dentro do processo
  - Para usar valor fora do processo, atribuir para um sinal
- Sinais são escalonados ao longo dos comandos do processo e só atribuídos no final

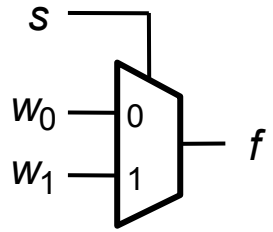
IC-UNICAMP

Processos: uso em circuitos digitais

- Principal aplicação
  - Descrição de circuitos sequenciais (a ser visto nas próximas aulas)
  - Implementação de funções complexas
- Mas também é possível implementar circuitos combinacionais

MUX 2:1 com if-then-else

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
    PORT (   w0, w1, s    : IN   STD_LOGIC ;
             f            : OUT  STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
    PROCESS ( w0, w1, s )
    BEGIN
        IF s = '0' THEN
            f <= w0 ;
        ELSE
            f <= w1 ;
        END IF ;
    END PROCESS ;
END Behavior ;
```



IC-UNICAMP

MUX 2:1 alternativo

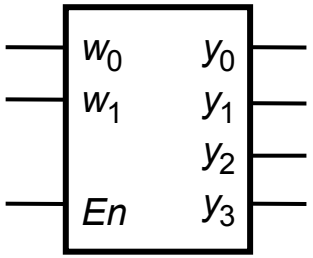
```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
   PORT (   w0, w1, s   : IN  STD_LOGIC ;
            f           : OUT STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
   PROCESS ( w0, w1, s )
   BEGIN
      f <= w0 ;
      IF s = '1' THEN
         f <= w1 ;
      END IF ;
   END PROCESS ;
END Behavior ;
```

IC-UNICAMP

MUX 2:1 com CASE

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY mux2to1 IS
    PORT (w0, w1, s    : IN     STD_LOGIC ;
          f        : OUT   STD_LOGIC ) ;
END mux2to1 ;

ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
    PROCESS ( w0, w1, s )
    BEGIN
        CASE s IS
            WHEN '0' =>
                f <= w0 ;
            WHEN OTHERS =>
                f <= w1 ;
        END CASE ;
    END PROCESS ;
END Behavior ;
```

Decodificador 2:4 – com processo (1)



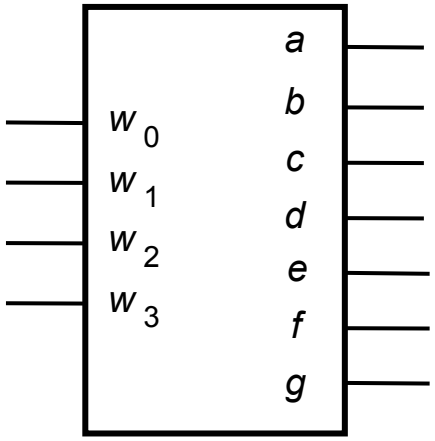

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY dec2to4 IS
   PORT (w  : IN  STD_LOGIC_VECTOR(1 DOWNTO 0) ;
         En : IN  STD_LOGIC ;
         y  : OUT STD_LOGIC_VECTOR(0 TO 3) ) ;
END dec2to4 ;
```

IC-UNICAMP

Decodificador 2:4 – com processo (2)

```
ARCHITECTURE Behavior OF dec2to4 IS
BEGIN
    PROCESS ( w, En )
    BEGIN
        IF En = '1' THEN
            CASE w IS
                WHEN "00" =>        y <= "1000" ;
                WHEN "01" =>        y <= "0100" ;
                WHEN "10" =>        y <= "0010" ;
                WHEN OTHERS =>      y <= "0001" ;
            END CASE ;
        ELSE
            y <= "0000" ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

BCD → 7 segmentos (1)




```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY seg7 IS
   PORT (bcd : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
         leds: OUT STD_LOGIC_VECTOR(1 TO 7) ) ;
END seg7 ;
ARCHITECTURE Behavior OF seg7 IS
```

IC-UNICAMP

BCD → 7 segmentos (2)

```
ARCHITECTURE Behavior OF seg7 IS
BEGIN
   PROCESS ( bcd )
   BEGIN
      CASE bcd IS   --       abcdefg
         WHEN "0000"    => leds     <=    "1111110" ;
         WHEN "0001"    => leds     <=    "0110000" ;
         WHEN "0010"    => leds     <=    "1101101" ;
         WHEN "0011"    => leds     <=    "1111001" ;
         WHEN "0100"    => leds     <=    "0110011" ;
         WHEN "0101"    => leds     <=    "1011011" ;
         WHEN "0110"    => leds     <=    "1011111" ;
         WHEN "0111"    => leds     <=    "1110000" ;
         WHEN "1000"    => leds     <=     "1111111" ;
         WHEN "1001"    => leds     <=     "1110011" ;
         WHEN OTHERS    => leds     <=     "-------" ;
      END CASE ;
   END PROCESS ;
END Behavior ;
```

Don´t care

IC-UNICAMP

Codificador prioridade (1)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
ENTITY priority IS
   PORT (   w      : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            y   : OUT     STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            z   : OUT     STD_LOGIC ) ;
END priority ;
```

IC-UNICAMP

Codificador prioridade (1)

```
ARCHITECTURE Behavior OF priority IS
BEGIN
    PROCESS ( w )
    BEGIN
        IF w(3) = '1' THEN
            y <= "11" ;
        ELSIF w(2) = '1' THEN
            y <= "10" ;
        ELSIF w(1) = '1' THEN
            y <= "01" ;
        ELSE
            y <= "00" ;
        END IF ;
    END PROCESS ;
    z <= '0' WHEN w = "0000" ELSE '1' ;
END Behavior ;
```

IC-UNICAMP

Outro codificador prioridade (1)

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY priority IS
   PORT (   w      : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            y   : OUT     STD_LOGIC_VECTOR(1 DOWNTO 0) ;
            z   : OUT     STD_LOGIC ) ;
END priority ;
```

IC-UNICAMP

Outro codificador prioridade (1)

```
ARCHITECTURE Behavior OF priority IS
BEGIN
   PROCESS ( w )
   BEGIN
      y <= "00" ;
      IF w(1) = '1' THEN y <= "01" ; END IF ;
      IF w(2) = '1' THEN y <= "10" ; END IF ;
      IF w(3) = '1' THEN y <= "11" ; END IF ;

      z <= '1' ;
      IF w = "0000" THEN z <= '0' ; END IF ;
   END PROCESS ;
END Behavior ;
```

IC-UNICAMP

Erros comuns no uso de processo

- Memória implícita
- Atribuição múltipla de um sinal dentro de um processo
- Feedback de sinal: oscilação

IC-UNICAMP

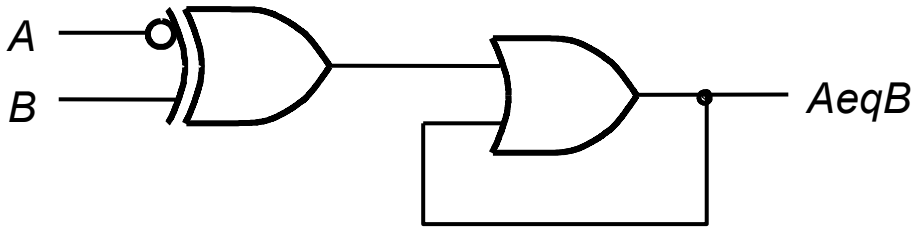
Problema: Comparador de 1 bit

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY implied IS
    PORT (      A, B   : IN      STD_LOGIC ;
                AeqB   : OUT STD_LOGIC ) ;
END implied ;

ARCHITECTURE Behavior OF implied IS
BEGIN
    PROCESS ( A, B )
    BEGIN
       IF A = B THEN
          AeqB <= '1' ;
       END IF ;
    END PROCESS ;
END Behavior ;
```

IC-UNICAMP

Problema: memória implícita

```
...
PROCESS ( A, B )
BEGIN
   IF A = B THEN
      AeqB <= '1' ;
   END IF ;
END PROCESS ;
 ...
```



IC-UNICAMP

Comparador de 1 bit corrigido

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY compare1 IS
   PORT (      A, B   : IN     STD_LOGIC ;
               AeqB   : OUT STD_LOGIC ) ;
END compare1 ;

ARCHITECTURE Behavior OF compare1 IS
BEGIN
   PROCESS ( A, B )
   BEGIN
      AeqB <= '0' ;
      IF A = B THEN
         AeqB <= '1' ;
      END IF ;
   END PROCESS ;
END Behavior ;
```

IC-UNICAMP

Contador de 1s

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY numbits IS
   PORT ( X  : IN  STD_LOGIC_VECTOR(1 TO 3) ;
              Count : BUFFER INTEGER RANGE 0 TO 3 ) ;
END numbits ;
```

Intenção: iniciar sinal Count com 0 e incrementar a cada ‘1’ encontrado no vetor X

Contador de 1s: 2 problemas

IC-UNICAMP

Realimentação → oscilação

```
ARCHITECTURE Behavior OF numbit
BEGIN
   PROCESS ( X ) - conta n° de      em X
   BEGIN
       Count <= 0 ; -- o 0      m aspas é decimal
       FOR i IN 1 TO 3 LOO
             IF X(i) = '1'  THEN
                   Count <= Count + 1 ;
             END IF ;
       END LOOP ;
   END PROCESS ;
END Behavior ;
```

Dupla atribuição de Count → só a última vai ser atribuída

IC-UNICAMP

Contador de 1s: corrigido

```
ARCHITECTURE Behavior OF Numbits IS
BEGIN
    PROCESS ( X ) -- conta n° de 1s em X
       VARIABLE TMP : INTEGER ;
   BEGIN
       Tmp := 0 ;
       FOR i IN 1 TO 3 LOOP
             IF X(i) = '1' THEN
                   Tmp := Tmp + 1 ;
             END IF ;
       END LOOP ;
       Count <= Tmp ;
   END PROCESS ;
END Behavior ;
```

IC-UNICAMP

Precedência das operações

Operator Class	Operator
Highest precedence Miscellaneous	**, ABS, NOT
Multiplying	*, /, MOD, REM
Sign	+, −
Adding	+, −, &
Relational	=, /=, <, <=, >, >=
Lowest precedence Logical	AND, OR, NAND, NOR, XOR, XNOR

- Dentro da mesma classe: da esquerda para a direita
  - a AND b OR c ----→ inválido
  - usar
    - `(a AND b) OR c` ou
    - `a AND (b OR c)`
- Recomendável: usar parênteses explicitamente

IC-UNICAMP

Construções de VHDL vistas nesta aula

- Selected Signal Assignment:

```
WITH ctl SELECT f <= w0 WHEN '0', w1 WHEN OTHERS
```

- Conceito de OTHERS na atribuição
- Atribuição condicional de sinal

```
f <= w0 WHEN ctl = '0' ELSE w3;
```

- Componentes e packages
- GENERATE
- **`OTHERS => '1'`**
- Tipo **`Signed`**
- Biblioteca **`std_logic_arith`**
- Vetores de bits: `STD_LOGIC_VECTOR(3 DOWNTO 0)`
- Generic

IC-UNICAMP

Construções de VHDL vistas nesta aula

- Conceito de processo e comandos sequenciais
- Construções internas ao processo
  - Variáveis
  - **`IF .. THEN ELSIF .. THEN ELSE ..`**
  - **`CASE .. IS WHEN .. => ..`**